Activities

What is an Activity?

- The major building block for all Views (UI) in an application
- Entry point for users to begin interacting with an app
- Used to manage user navigation through an app
- Provides mechanisms for common events such as
  - Managing orientation changes in a manner that doesn't disrupt the user experience
  - Ensuring that user data remains stateful and portable from View to View (read: Activity to Activity)
  - Managing termination of stale or completed processes

Application Views

- Activities provides an aperture for the Android Framework to draw the User Interface.
  - Typically (but not always) full screen
  - Extends the Activity class (or a child of the Activity class such as AppCompatActivity)
- As a rule of thumb, one Activity is used for each view (or 'screen') within an app.
  - Apps often contain multiple views, which means that they are comprised of multiple activities.

Main Activity

- All apps must designate a 'main activity' which serves as the first view that a user see when entering the app
- The main activity can then serve as a navigation point for all other activities (views) in the app
- The relationship between the main activity (and in fact between all activities) is a loosely bound one meaning that there are minimal dependencies.
  - This means that well designed Activities should ideally be 'started' or navigated to by any other Activity that satisfies it's dependencies.
- The main activity is designated in the ApplicationManifest.xml

Activities and the Application Manifest

- In order for an application to use Activities, they must be added to the Application Manifest (ApplicationManifest.xml)
  - Activities must be declared as child <activity> elements under the <application> tag
  - Each element must, at minimum, be provided with an 'android:name' attribute
    - The name attribute should be directed toward the namespace and classname of the the activity being registered

Activities and the Application Manifest

- The <activity> element can have an <intent-filter> child element (Intents and various filters to be covered later in this lecture)
  - The <intent-filter> element can contain a number of child elements that describe how the Activity being registered should behave at a global level. These child elements can be of different types (action, category, data) which we will cover more in depth at a later time.
  - The two types that we care about for the purposes of registering the main activity are the <action> and <category> types

LAUNCHER and MAIN

- The first child element in the <intent-filter> collection that we need to declare is

  <action android:name="android.intent.action.MAIN" />

  This filter is of the <action> variety and tells the Android Framework that when the application is launched, the registered activity should be the point of entry for the application.

LAUNCHER and MAIN

- The next child element in the <intent-filter> collection that we need to declare is

  <category android:name="android.intent.category.LAUNCHER" />

  This filter is of the <category> variety and tells the Android Framework that the activity should be registered in the top level app launcher.

Activity Lifecycle

Activity Lifecycle

- Each activity has it's own lifecycle.
- This lifecycle is managed by a set of methods implemented in the Activity class.
  - These methods are called by the Android OS, not the developer
- These methods can also be overridden in the implementing class to enable special handling.

Activity Lifecycle

- onCreate
- onStart

- onPause
- onResume

- onStop
- onDestroy

Lifecycle Method: onCreate

- Executed when the Activity is being created.
- Typically contain code for setting the contextual link between the activity and the layout (setContentView).
- Generally takes care of the getting everything ready for app interaction (UI, graphics, sound)

Lifecycle Method: onStart

- Executes with the app is in the starting phase.
- Always follows onCreate
- Runs when the Activity begins to become visible to the user.
- Typically do things that should be synced with visibility
  - e.g. music: *load* the music in onCreate, *play* the music in onStart

Lifecycle Method: onPause/onResume

- Executed after onStart
- The onResume method is executed only if the Activity has been resumed after being paused. This might take the form of reloading previously saved user data from when the app had been interrupted (e.g. by a phone call)
- The onPause method is executed when the app is paused or backgrounded (e.g. by a phone call). Execution of this method typically contains logic to save relevant unsaved data so that the state of the Activity can be reloaded in onResume.

Lifecycle Method: onStop

- As the name implies, the execution of this method is related to the stopping of the Activity/app.
- Possibly undoes everything done in onCreate (releasing resources, writing information to a database).
- Entry into this method typically indicates that the Activity is going to be destroyed

Lifecycle Method: onDestroy

- As the name implies, this method is entered when the activity is actually being destroyed.

Android Intent Object

What is an Intent

- An Intent is a class that demonstrates the 'intent' of an Activity in an app. It makes clear intent and also facilitates it.
- In apps containing more than one Activity, the Intent object allows us to switch between Activities.
- Since Activities are individual classes, interoperable data for those activities can be passed via Intent.
- Intents may also be used to interact with other apps. As an example, an Intent could be used to facilitate links in an app that send email, make phone calls or open web browsers.